

Klasifikasi Dokumen Berita Kejadian Berbahasa Indonesia dengan Algoritma Single Pass Clustering

Agus Zainal Arifin dan Ari Novan Setiono

Jurusan Teknik Informatika, Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember (ITS) – Surabaya
Kampus ITS, Jl. Raya ITS, Sukolilo-Surabaya 60111
Tel. +62 31 5939214, Fax + 62 31 5939363
Email : agusza@its-sby.edu

Abstrak

Pengelompokan dokumen berita dibutuhkan untuk mempermudah pencarian informasi mengenai suatu *event* (kejadian) tertentu. Pencarian berita-berita lain yang berkaitan dengan kejadian tersebut tentu sulit dilakukan, bila hanya mengandalkan *query* biasa. Sebab pemilihan *query* yang kurang spesifik akan berakibat membanjirnya dokumen-dokumen yang tidak relevan.

Penelitian ini berusaha untuk mengklasifikasi dokumen dengan menggunakan algoritma *single pass clustering*. Klasifikasi ini ditekankan untuk dokumen berita berbahasa Indonesia, sebab dewasa ini, kebutuhan konsumen di tanah air terhadap informasi semakin meningkat. Adapun keterkaitan antar berita ini dapat diukur berdasarkan kemiripan antar dokumen (*similarity*).

Algoritma ini diuji coba dengan menggunakan sampel berita dari media massa berbasis web. Hasil uji coba menunjukkan bahwa algoritma ini dapat diaplikasikan untuk pengelompokan berita-berita berbahasa Indonesia. Pemilihan nilai *threshold* yang tepat akan meningkatkan kualitas *information retrieval* (temu kembali informasi) pada dokumen. Kualitas ini tercermin dari tingkat *recall* 79 % dan *precision* 88 %.

Kata Kunci : *Information Retrieval, Stemming, Single Pass Clustering, Cosine Similarity, event classification*

I. Pendahuluan

Dengan semakin berkembangnya Teknologi Informasi yang dibutuhkan oleh pengguna mengakibatkan munculnya suatu cabang ilmu baru dalam Teknologi Informasi yaitu Pencarian Informasi (*Information Retrieval*). Pencarian Informasi berbasis *query* (*Query based retrieval*) yang digunakan secara tradisional sangat berguna untuk pencarian terarah tetapi tidak begitu efisien untuk *generic query* [1].

Query based Retrieval berguna ketika kita mengetahui benar kejadian asli atau fakta yang dicari. Cara ini tidak begitu efisien ketika kita membutuhkan informasi yang spesifik/khusus dalam kategori yang besar.

Cara ini tidak begitu efisien untuk mendapatkan berita yang relevan dalam penelusuran suatu kejadian. Dengan menggunakan cara ini biasanya hanya dapat diketahui berita-berita tertentu saja, sedangkan berita-berita lama yang berkaitan dengan berita tersebut sulit untuk ditelusuri.

Yang diperlukan dalam suatu Sistem yang cerdas (*Intelligent System*) adalah secara otomatis sistem ini dapat :

- mendeteksi kejadian penting dari sekian banyak berita-berita baru

- Memasukkan berita-berita kejadian baru pada kelompok berita yang sudah ada.
- Membuat kelompok berita baru apabila suatu berita tidak memenuhi kelompok berita yang sudah ada
- Menampilkan kejadian yang diperlukan oleh user dalam bentuk berita-berita terkait yang ada hubungannya dengan keinginan user [1].

Inilah yang menjadi tujuan utama dari penelitian yang *Topic Detection and Tracking (TDT)*. Topik (*Topic*) disini adalah perubahan kejadian secara dinamis. Dalam metode ini kita menggunakan beberapa cara untuk pencarian informasi (*Information Retrieval*) dan teknik *machine learning* untuk keefektifan pendeteksian dan klasifikasi(detection) [3]

Pendeteksian berita dalam *Topic Detection and Tracking (TDT)* merupakan suatu cara untuk mendapatkan suatu solusi dalam pencarian informasi (*Information Retrieval*). Dari permasalahan yang mungkin muncul inilah maka dikembangkan suatu metode untuk menangani masalah-masalah di atas, dalam hal ini contoh kasus yang diangkat adalah pendeteksian dan pengelompokan suatu kejadian yang merupakan bagian dari *Event Detecting* dalam TDT. Pada dasarnya TDT dapat dibagi menjadi 2 tahap :

1. **Penyiapan Dokumen**

Pada tahap ini akan dilakukan manipulasi teks pada dokumen yang selanjutnya tiap dokumen akan direpresentasikan dalam bobot tertentu

2. **Klasifikasi dokumen**

Untuk klasifikasi dokumen diperlukan nilai batas (*Threshold value*). Untuk mendapatkan nilai batas (*Threshold value*) diperlukan suatu data training (*restrospective document*).

Permasalahan yang timbul dari klasifikasi dokumen berita adalah :

1. Perlunya algoritma klasifikasi dokumen
2. Penentuan tingkat kemiripan (*similarity*) antar dokumen berdasarkan komposisi term
3. Pemilihan *term* dari kata-kata dalam Bahasa Indonesia yang relevan sebagai pembeda
4. Belum tersedianya kamus kata dasar dalam bahasa Indonesia sebagai acuan penentuan *term*.

II. Metodologi

1. **Pengumpulan data sampel**

Dokumen yang digunakan sebagai sampel adalah berita-berita yang diambil dari Surat Kabar *Online* Suara Pembaharuan Tanggal 1 Agustus 2001 sampai dengan tanggal 31 Agustus 2001. Jumlah sampel adalah 86 dokumen berita yang diklasifikasikan secara manual menjadi 23 *event*. Klasifikasi manual ini nantinya digunakan untuk mengevaluasi tingkat keberhasilan klasifikasi.

2. **Ekstraksi dokumen**

Proses ekstraksi ini bertujuan untuk menghasilkan *term-term* yang akan digunakan sebagai *prototype* bagi setiap dokumen. Tiap *term* tersebut dicari bentuk kata dasarnya berdasarkan kamus kata dasar Bahasa Indonesia. Hal ini untuk menghindari tersimpannya kata-kata yang memiliki kata dasar yang sama namun berimbuhan berbeda. Disamping itu dilakukan penyaringan (*filtering*) terhadap kata-kata yang tidak layak untuk dijadikan sebagai pembeda. Kelompok kata ini biasanya disebut sebagai *stoplist*. Oleh karena belum tersedia maka penelitian ini juga berusaha mencari *stoplist* tersebut secara manual.

3. **Penghitungan Bobot TF-IDF**

Pada tahap ini, tiap dokumen diwujudkan sebagai sebuah *vector* dengan elemen sebanyak *term* yang berhasil

dikenali dari tahap ekstraksi dokumen di atas. Vektor tersebut beranggotakan bobot dari tiap *term* yang dihitung berdasarkan metode TF-IDF. Metode TF-IDF ini merupakan metode pembobotan dalam bentuk sebuah metode yang merupakan integrasi antar *term frequency (tf)*, dan *inverse document frequency (idf)* [1][9]. Adapun rumusnya adalah :

$$w(t,d) = tf(t,d) * \log_2(N/n_d)$$

Simbol $w(t,d)$ adalah bobot dari *term t* dalam dokumen *d* sedangkan $tf(t,d)$ adalah frekuensi *term* dalam dokumen (*tf*) dimana *N* merupakan ukuran data training yang digunakan untuk penghitungan IDF. Adapun n_d adalah jumlah dari dokumen yang di-training yang mengandung nilai *t*.

Fungsi metode ini adalah untuk mencari representasi nilai dari tiap-tiap dokumen dari suatu kumpulan data training (*training set*). Dari sini akan dibentuk suatu vektor antara dokumen dengan kata (*documents with terms*) yang kemudian untuk kesamaan antar dokumen dengan cluster akan ditentukan oleh sebuah *prototype* vektor yang disebut juga dengan *cluster centroid* [1].

4. **Penghitungan tingkat kemiripan**

Perbandingan kemiripan (*similarity*) yang digunakan disini adalah *standard cosine similarity* [5] [6] dengan rumus :

$$S_{D_i, D_j} = \frac{\sum_{k=1}^L (weight_{ik} weight_{jk})}{\sqrt{\sum_{k=1}^L weight_{ik}^2 \sum_{k=1}^L weight_{jk}^2}}$$

S_{D_i, D_j} : Similarity Dokumen ke I dan ke j

5. **Klasifikasi**

Similarity yang telah dihasilkan selanjutnya dievaluasi untuk menentukan pasangan-pasangan dokumen yang dinyatakan mirip berdasarkan nilai *threshold* tertentu. Pengklasifikasian dokumen berita dengan menggunakan Algoritma *Single Pass Clustering*.

6. **Evaluasi klasifikasi**

Evaluasi ini dilakukan untuk mengetahui kinerja algoritma klasifikasi pada tahap uji coba. Pengukuran ini didasarkan pada dua parameter, yakni *recall* dan *precision*.

III. Stemming Bahasa Indonesia

Algoritma ini didahului dengan pembacaan tiap kata dari file sampel. Sehingga input dari algoritma ini adalah sebuah kata yang kemudian dilakukan :

1. Pemeriksaan semua kemungkinan bentuk kata. Setiap kata diasumsikan memiliki 2 Awalan (*prefiks*) dan 3 Akhiran (*sufiks*).

- Sehingga bentuknya menjadi :
Prefiks 1 + Prefiks 2 + Kata dasar + Sufiks 3 + Sufiks 2 + Sufiks 1
 Seandainya kata tersebut tidak memiliki imbuhan sebanyak imbuhan di atas, maka imbuhan yang kosong diberi tanda x untuk prefiks dan diberi tanda xx untuk sufiks.
2. Pemotongan dilakukan secara berurutan sebagai berikut :
 AW : AW
 AK : AK
 KD : KD
 - a. AW I, hasilnya disimpan pada p1
 - b. AW II, hasilnya disimpan pada p2
 - c. AK I, hasilnya disimpan pada s1
 - d. AK II, hasilnya disimpan pada s2
 - e. AK III, hasilnya disimpan pada s3
 Pada setiap tahap pemotongan di atas diikuti dengan pemeriksaan di kamus apakah hasil pemotongan itu sudah berada dalam bentuk dasar. Kalau pemeriksaan ini berhasil maka proses dinyatakan selesai dan tidak perlu melanjutkan proses pemotongan imbuhan lainnya
 Contoh pemenggalan kata "mempermainkannya"
 Langkah 1 :
 Cek apakah kata ada dalam kamus
 Ya : Success
 Tidak : lakukan pemotongan AW I
 Kata = permainkannya
 Langkah 2 :
 Cek apakah kata ada dalam kamus
 Ya : Success
 Tidak : lakukan pemotongan AW II
 Kata = mainkannya
 Langkah 3 :
 Cek apakah kata ada dalam kamus
 Ya : Success
 Tidak : lakukan pemotongan AK I
 Kata = mainkan
 Langkah 4 :
 Cek apakah kata ada dalam kamus
 Ya : Success
 Tidak : lakukan pemotongan AK II
 Kata = main
 Langkah 5 :
 Cek apakah kata ada dalam kamus
 Ya : Success
 Tidak : lakukan pemotongan AK III. Dalam hal ini AK III tidak ada, sehingga kata tidak diubah.
 Kata = main
 Langkah 6
 Cek apakah kata ada dalam kamus
 Ya : Success
 Tidak : "Kata tidak ditemukan"

3. Namun jika sampai pada pemotongan AK III, belum juga ditemukan di kamus, maka dilakukan proses kombinasi. KD yang dihasilkan dikombinasikan dengan imbuhan-imbuhan dalam 12 konfigurasi berikut :
 - a. KD
 - b. KD + AK III
 - c. KD + AK III + AK II
 - d. KD + AK III + AK II + AK I
 - e. AW I + AW II + KD
 - f. AW I + AW II + KD + AK III
 - g. AW I + AW II + KD + AK III + AK II
 - h. AW I + AW II + KD + AK III + AKII + AKI
 - i. AW II + KD
 - j. AW II + KD + AK III
 - k. AW II + KD + AK III + AK II
 - l. AW II + KD + AK III + AK II + AK I

Sebenarnya kombinasi a, b, c, d, h, dan l sudah diperiksa pada tahap sebelumnya, karena kombinasi ini adalah hasil pemotongan bertahap tersebut. Dengan demikian, kombinasi yang masih perlu dilakukan tinggal 6 yakni pada kombinasi-kombinasi yang belum dilakukan (e, f, g, i, j, dan k). Tentunya bila hasil pemeriksaan suatu kombinasi adalah 'ada', maka pemeriksaan pada kombinasi lainnya sudah tidak diperlukan lagi.

Pemeriksaan 12 kombinasi ini diperlukan, karena adanya fenomena overstemming pada algoritma pemotongan imbuhan. Kelemahan ini berakibat pada pemotongan bagian kata yang sebenarnya adalah milik Kata dasar itu sendiri yang kebetulan mirip dengan salah satu jenis imbuhan yang ada. Dengan 12 kombinasi itu, pemotongan yang sudah terlanjur tersebut dapat dikembalikan sesuai posisinya.

IV. Algoritma *Single Pass Clustering*

Algoritma Single Pass Clustering [5] dapat dilakukan dengan langkah-langkah sebagai berikut :

1. Masukkan (dokumen pertama) D1 representasi (*Cluster* pertama) C1
2. Untuk (dokumen ke-i) Di hitung kesamaan (*similarity*) dengan setiap wakil dari masing-masing *cluster*.
3. Jika (*Maximum Similarity*) S_{max} lebih besar dari batas nilai (*threshold value*) S_T , tambahkan tambahkan item kepada cluster yang bersesuaian dan hitung kembali representasi *cluster*, sebaliknya gunakan Di untuk inialisasi *cluster* baru.
4. Jika masih ada sebuah item Di yang belum dikelompokkan, kembali ke langkah ke-2

Algoritma ini digunakan untuk retrospective data detection maupun on-line detection

V. Uji Coba

Untuk mengevaluasi secara manual kesamaan diantara dokumen dalam cluster-cluster yang telah dikelompokkan digunakan standar sebagaimana Tabel 2. Tabel tersebut berisi berbagai kemungkinan hasil klasifikasi pada tiap event (*Per Event contingency table*) [1].

Tabel 2. kategori hasil klasifikasi

	In Event	Not In event
<i>In cluster</i>	a	b
<i>Not In Cluster</i>	c	d

Tabel 2 menunjukkan bahwa hasil klasifikasi adakalanya memang termasuk event (a) yang dimaksud dan adakalanya tidak (b). Sedangkan dokumen yang tidak termasuk dalam hasil klasifikasi suatu event, adakalanya memang bukan anggota event itu (d) dan adakalanya ternyata seharusnya menjadi anggota event tersebut (c). Dalam hal ini, keempat parameter di atas digunakan untuk menghitung 2 parameter evaluasi, yakni :

1. *Recall*, yakni tingkat keberhasilan mengenali suatu event dari seluruh event yang seharusnya dikenali. Rumusnya adalah $r = a/(a+c)$ untuk $a+c > 0$. Selain itu tidak didefinisikan
2. *Precision*, yakni tingkat ketepatan hasil klasifikasi terhadap suatu event. Artinya, dari seluruh dokumen hasil klasifikasi, berapa persenkah yang dinyatakan benar. Rumusnya adalah $p = a/(a+b)$ jika $a+b > 0$. Selain itu tidak didefinisikan

Dari hasil evaluasi yang dilakukan terhadap data training yang diambil dari suara pembaruan *online* mulai tanggal 1 Agustus 2001 sampai dengan 31 Agustus 2001 dengan perincian sebagai berikut :

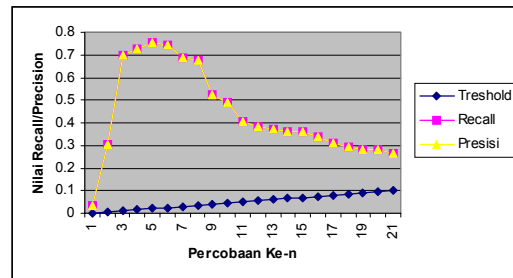
- Diambil 86 Dokumen sebagai Restrospective/training set yang dikalsifikasikan kedalam 23 event
- Setelah melalui proses *stemming* maka dapat dilakukan penghitungan frekuensi kata dalam dokumen dengan menggunakan kamus sejumlah 29.349 kata dasar bahasa Indonesia
- Dari matrik yang dibentuk dihasilkan sebanyak 13.000 *record* untuk kaitan dokumen dengan kata $tf(t,d)$ dengan frekuensi di atas 0

Didapatkan hasil nilai *Recall* dan *Precision* sebagaimana Tabel 1.

Tabel 1. Hasil uji coba

Percobaan Ke-	Treshold	Recall	Precision
1	0	0.034883721	0.034883721
2	0.005	0.302325581	0.302325581
3	0.01	0.697674419	0.697674419
4	0.015	0.727272727	0.727272727
5	0.02	0.755813953	0.755813953
6	0.025	0.744186047	0.744186047
7	0.03	0.686046512	0.686046512
8	0.035	0.674418605	0.674418605
9	0.04	0.523255814	0.523255814
10	0.045	0.488372093	0.488372093
11	0.05	0.406976744	0.406976744
12	0.055	0.38372093	0.38372093
13	0.06	0.372093023	0.372093023
14	0.065	0.360465116	0.360465116
15	0.07	0.360465116	0.360465116
16	0.075	0.337209302	0.337209302
17	0.08	0.30952381	0.30952381
18	0.085	0.290697674	0.290697674
19	0.09	0.279069767	0.279069767
20	0.095	0.279069767	0.279069767
21	0.1	0.26744186	0.26744186

Distribusi nilai kedua parameter dapat digambarkan dengan grafik sbgn gb1



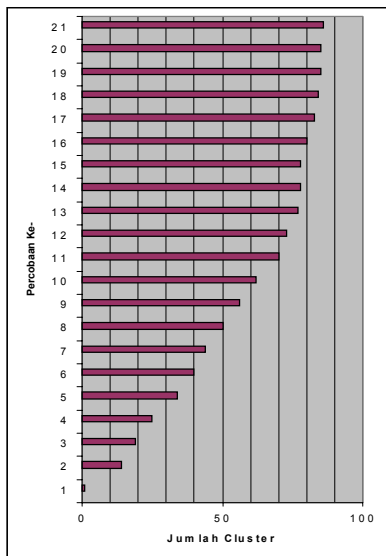
Gambar 1
Grafik Recall dan Precision untuk Algoritma Single Pass dg Microaverage

Dari sini terlihat bahwa nilai *Treshold* untuk hasil terbaik didapatkan pada angka 0.02. *treshold* ini didapatkan dari percobaan yang dilakukan secara linear terhadap 21 *treshold* yang berbeda. Percobaan dilakukan mulai dari nilai *treshold* = 0 sampai dengan mendapatkan jumlah *cluster* = jumlah dokumen atau nilai *treshold* di atas keseluruhan *similarity* maksimal

Sedangkan untuk jumlah cluster yang dihasilkan dari 21 percobaan dapat dilihat dalam Tabel 2 sebagai berikut :

Percobaan Ke-	Threshold	Jumlah Cluster
1	0	1
2	0.005	14
3	0.01	19
4	0.015	25
5	0.02	34
6	0.025	40
7	0.03	44
8	0.035	50
9	0.04	56
10	0.045	62
11	0.05	70
12	0.055	73
13	0.06	77
14	0.065	78
15	0.07	78
16	0.075	80
17	0.08	83
18	0.085	84
19	0.09	85
20	0.095	85
21	0.1	86

Distribusi jumlah cluster yang dihasilkan bisa digambarkan dengan gb 2 :



Gambar 2
Jumlah Cluster

Dari sini dapat disimpulkan bahwa hasil percobaan terbaik adalah dengan menggunakan threshold 0.02 walaupun dengan jumlah cluster yang terbentuk lebih banyak namun nilai *Recall* dan *Precision*-nya sangat tinggi

VI. Pembahasan

- Pada percobaan dengan menggunakan Algoritma *Single Pass Clustering* ini percobaan dimulai dengan *threshold* 0.

Pada *threshold* 0 ini akan menghasilkan 1 cluster. Percobaan akan diakhiri apabila menghasilkan jumlah *cluster* = jumlah dokumen

- Pada percobaan yang dilakukan nilai *Recall* dan *Precision* tertinggi dihasilkan pada *threshold* 0.02 dengan nilai *recall* **0.755813953** dan menghasilkan 34 *cluster*.
- Dari percobaan ini akan dilakukan lagi dengan menggunakan *threshold* antara terdekat antara 0.015 dengan 0.02 yaitu 0.0175 dan menghasilkan nilai *recall-precision* 0.790698(79 %) dan jumlah cluster 29 dan juga dengan menggunakan *threshold* 0.0225 menghasilkan *recall-precision* 0.744186(74 %). Dengan memakai *threshold* 0.0175 didapatkan *recall* rata-rata 0.764372 dan *precision* rata-rata 0.877536
- Dengan demikian *Algoritma Single Pass Clustering* cukup handal untuk digunakan untuk klasifikasi

VI. Kesimpulan dan Saran

1. Kesimpulan

- *Stemming* Bahasa Indonesia yang digunakan efektif untuk memilih *prototype* kata dasar yang digunakan sebagai pembeda antar dokumen
- Pembobotan TF IDF dan *Cosine Similarity* digunakan untuk menunjukkan kemiripan antar dokumen
- Berita yang mempunyai event yang sama cenderung untuk mengelompok menjadi 1 cluster.
- Nilai *threshold*(nilai batas) yang paling bagus digunakan adalah 0.0175 dengan nilai *recall-precision* 79 % dan nilai *recall* rata-rata 76 % dan *precision* rata-rata 87 %
- *Single Pass clustering* cukup handal digunakan sebagai algoritma untuk klasifikasi *event*

2. Saran

- Perlu adanya riset lebih mendalam untuk membedakan AK -an dan -kan dalam stemming bahasa Indonesia
- Perlu perbandingan dengan algoritma lain untuk mencapai hasil yang lebih optimal
- Untuk term yang bernilai 0 dalam setiap dokumen tidak perlu dilibatkan dalam perhitungan karena hanya akan menambah waktu perhitungan

- Pembuatan *stoplist* bisa dilakukan secara otomatis (*increment*) dengan menggunakan algoritma IDF

VII. DAFTAR PUSTAKA

- 1 Yiming Yang, Jaime G. Carbonell, Rulf D. Brown, Thomas Pierce, Brian T. Achibald, Xin Liu. "*Learning Approaches for Detecting and Tracking News Events*". IEEE Intelligent Systems, Language Technologies Institute, Carbegie Mellon University, 1999
- 2 Ron Papka. "*On-Line New Event Detection, Clustering, and Tracking*". Ph. D dissertation on University of Massachusetts, 1999
- 3 J. Allan et al, "*Topic Detection and Tracking Pilot Study : Final Report.*" Proc. DARPA Broadcast News Transcription & Understanding Workshop, Morgan Kaufman, San Francisco, 1998, pp194-218
- 4 Ricardo Baeza-Yates, Berthier Ribeiro-Neto, "*Modern Information Retrieval*" ACM Press, New York, Addison Wasley Longman Limited, 1999
- 5 William B. Frakes, Richardo Baeza-Yates, "*Information Retrieval Data Structures And Algorithm*", Prentice-Hall International Edition, 1992
- 6 Anil K. Jain, Richard C. Dubes, "*Algorithms for Clustering Data*", Prentice-Hall Advances Reference Series, 1988
- 7 Pakana, Fitrio "*Perancangan Dan Pembuatan Aplikasi Pencarian Dokumen Berbasis Web Dengan Penerapan Metode Suffix Tree Clustering Pada Result Set*", Tugas Akhir, Teknik Informatika, Institut Teknologi Sepuluh Nopember Surabaya, 2001
- 8 Weiss, Mark Allen, "*Data Structures & Problem Solving Using Java*", Florida International University, Addison Wesley, 1999
- 9 Salton, G.. Automatic text processing. Chapter 9, 1989